



Criação de Jogo do Gênero Puzzle



gora que preparamos o Unity com algumas configurações Prefers e Scripts Básicos iremos começar o projeto do game. Para tanto, nessa aula, iremos falar sobre:

- **Efeito elástico (corda da catapulta):** aprender a criar uma Linha Renderer para seguir o personagem;

- **Efeito Catapulta:** saber criar efeitos e códigos para arremessar o personagem;

- **Efeitos de Câmera:** entender e aprender como fazer a câmera seguir o personagem;

- **Morte do Personagem:** criar um código para desencadear a morte do personagem depois que atirado.

- **Efeito elástico**

Antes de começarmos a fazer os efeitos, é importante que você tenha instalado no seu computador o Unity Remote, para que você possa ver em tempo real o resultado do seu jogo no seu smartphone. Indico você assistir aos vídeos que explicam o passo-a-passo de como fazer:

- **Vídeo 1 - Configuração Unity Hub:** https://youtu.be/svC9uEd1xC0

- **Vídeo 2 - Configuração MOBILE:** https://youtu.be/OuJlc14vz_U



Para preparar o efeito estilingue será necessário que você prepare o seu Prefab “spartano” primeiro. Para tanto ele precisa estar com a seguinte configuração com um “Circle Collider 2D” e “Rigdbody 2D”, porém esse último com o BodyType como Kinematic e o Simulated selecionado.

Depois você precisará colocar nele um script C#, eu nomeei como “Drag”, e ele consistiu em:

- Criar as variáveis (Figura 1):

a. private Collider2D drag: tipo de colisor usado para a jogabilidade 2D;

b. public LayerMask layer; vai especificar o Layer onde a ação deve acontecer;

c. SerializeField: força o Unity a serializar um campo privado, ou seja, faz ela aparecer no inspector;

d. private bool clicked: torna o clicked em um objeto booleano, ou seja, verdadeiro ou falso;

e. private Touch touch: descrevendo o status de um dedo tocando a tela.

```
//variáveis
private Collider2D drag;
public LayerMask layer;
[SerializeField]
private bool clicked;
private Touch touch;
```

Figura 1 – Variáveis | Autor, 2022

2. Em Start (Figura 2) você indicará que drag é um objeto relacionado com o Collider2D no Unity;



```
Mensagem do Unity | 0 referências
void Start()
{
    drag = GetComponent<Collider2D>();
}
```

Figura 2 – Start | Autor, 2022

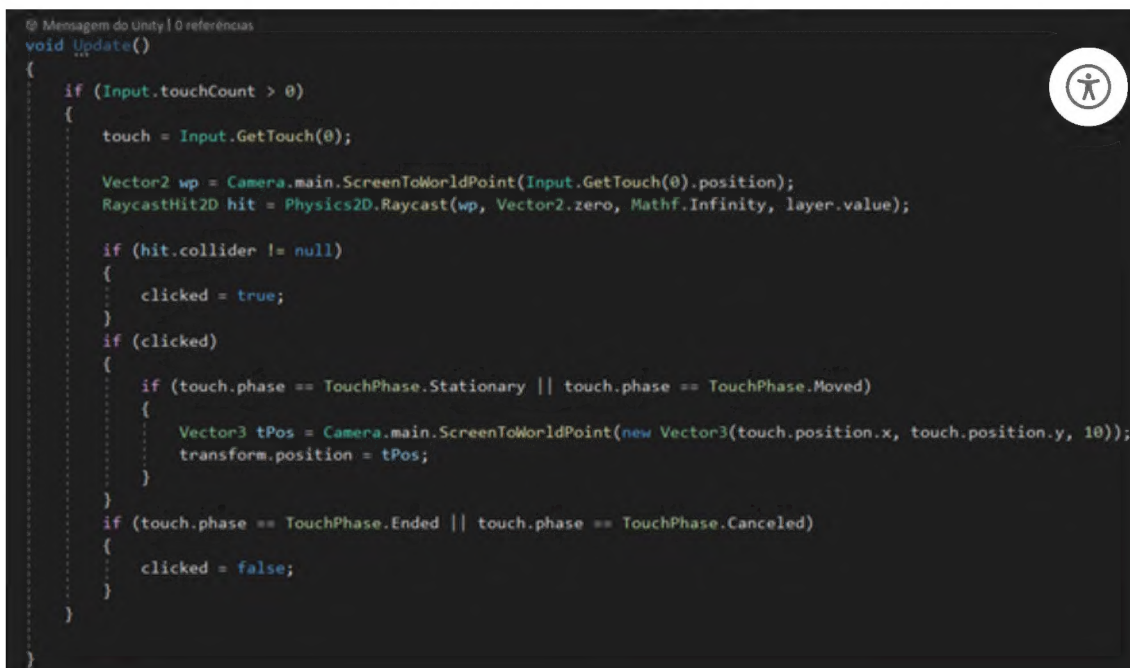
3. Já no Update (Figura 3):

a. você precisará chamar Input.GetTouch para obter uma estrutura de toque.

b. depois será necessário transformar um ponto do espaço da tela em espaço do mundo, nesse caso “o espaço do mundo” é definido como o sistema de coordenadas no topo da hierarquia do seu jogo, ou seja, na tela do smartphone.

c. O ponto do espaço do mundo criado pela conversão do ponto do espaço da tela na distância fornecida zero do painel da câmera para movimentos nos eixos “x” e “y”.

d. Em seguida você colocará a condição que fará com que quando seu dedo segurar ou movimentar o objeto, ele siga o movimento do seu dedo.



```
Mensagem do Unity | 0 referências
void Update()
{
    if (Input.touchCount > 0)
    {
        touch = Input.GetTouch(0);

        Vector2 wp = Camera.main.ScreenToWorldPoint(Input.GetTouch(0).position);
        RaycastHit2D hit = Physics2D.Raycast(wp, Vector2.zero, Mathf.Infinity, layer.value);

        if (hit.collider != null)
        {
            clicked = true;
        }
        if (clicked)
        {
            if (touch.phase == TouchPhase.Stationary || touch.phase == TouchPhase.Moved)
            {
                Vector3 tPos = Camera.main.ScreenToWorldPoint(new Vector3(touch.position.x, touch.position.y, 10));
                transform.position = tPos;
            }
        }
        if (touch.phase == TouchPhase.Ended || touch.phase == TouchPhase.Canceled)
        {
            clicked = false;
        }
    }
}
```

Figura 3 – Update | Autor, 2022

Agora é a vez de organizar o cenário e a catapulta. Podemos fazer isso arrastando as sprites (Figura 4-a) para a cena organizando-as por meio do Inspector>Order in Layer. Cada sprite, supondo que tenhamos 5, deverá ser numerada de forma que a 0 ficará atrás e 4 ficará na frente (Figura 4-b).

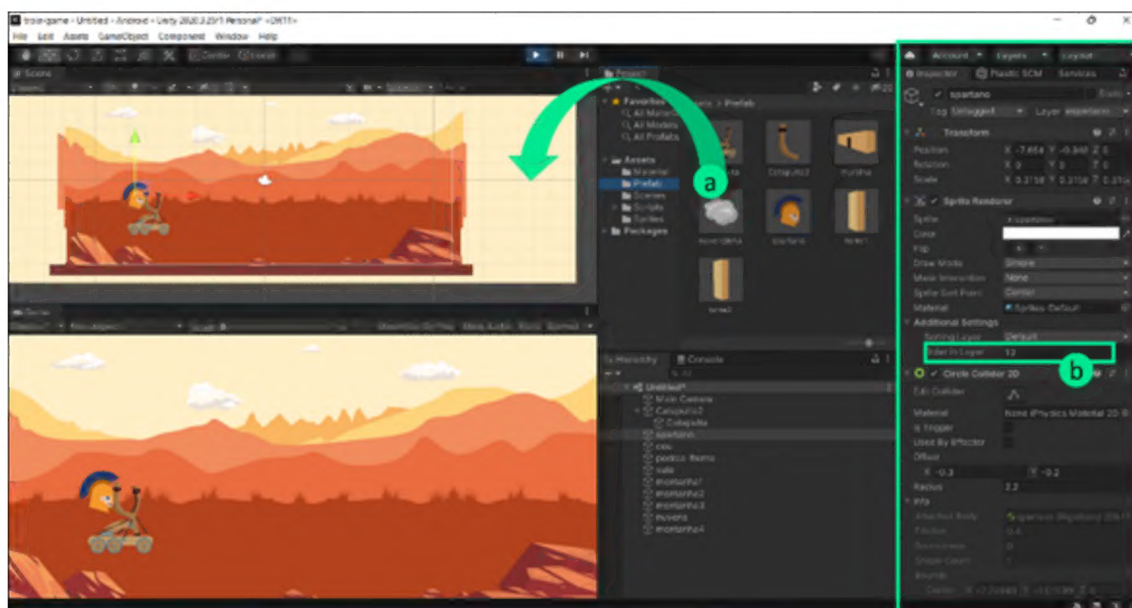


Figura 4 – Organização de Sprites e Prefabs na Layer | Autor, 2022

Depois disso será necessário selecionar as partes da Catapulta e Adicionar Componente> Effects > Line Renderer. Em seguida insira um material nelas e

agora é hora de programar. Abra o arquivo C# "Drag".



Crie agora mais duas variáveis para as Line Renderes (Figura 5).

```
//variáveis
private Collider2D drag;
public LayerMask layer;
[SerializeField]
private bool clicked;
private Touch touch;

public LineRenderer lineFront;
public LineRenderer lineBack;
```

Figura 5 – Line Renderer Catapulta

Após isso é a vez de você criar duas variantes "void" que serão responsáveis por fazer a Line Renderer (Tabela 1) seguir o personagem.

Primeira variante	Segunda variante
void SetupLine () { lineFront.SetPosition(0, lineFront.transform.position); lineBack.SetPosition(0, lineBack.transform.position); }	void LineUpdate () { lineFront.SetPosition(1, transform.position); lineBack.SetPosition(1, transform.position); }
Essa variante configurará a linha o lugar onde ela se originará, ou seja, do ponto "0".	Essa outra configura o comportamento da linha, ou seja, como ela se atualizará conforme o movimento do objeto ao qual o código está vinculado (personagem spartano).

Tabela 1 – Código C# Catapulta | Adaptado de NASCIMENTO, 2022

Para esses códigos funcionarem você precisará de duas coisas:

1. Ao selecionar o personagem, que possui o código C# (Figura 6-a), vincular na Unity o asset que contém as Lines Renderers, ou seja, os Prefabs referentes a catapulta (Figura 6-b e 6-c);

2. Inserir em “Start” o “SetupLine();” e em Update “LineUpdate();”.

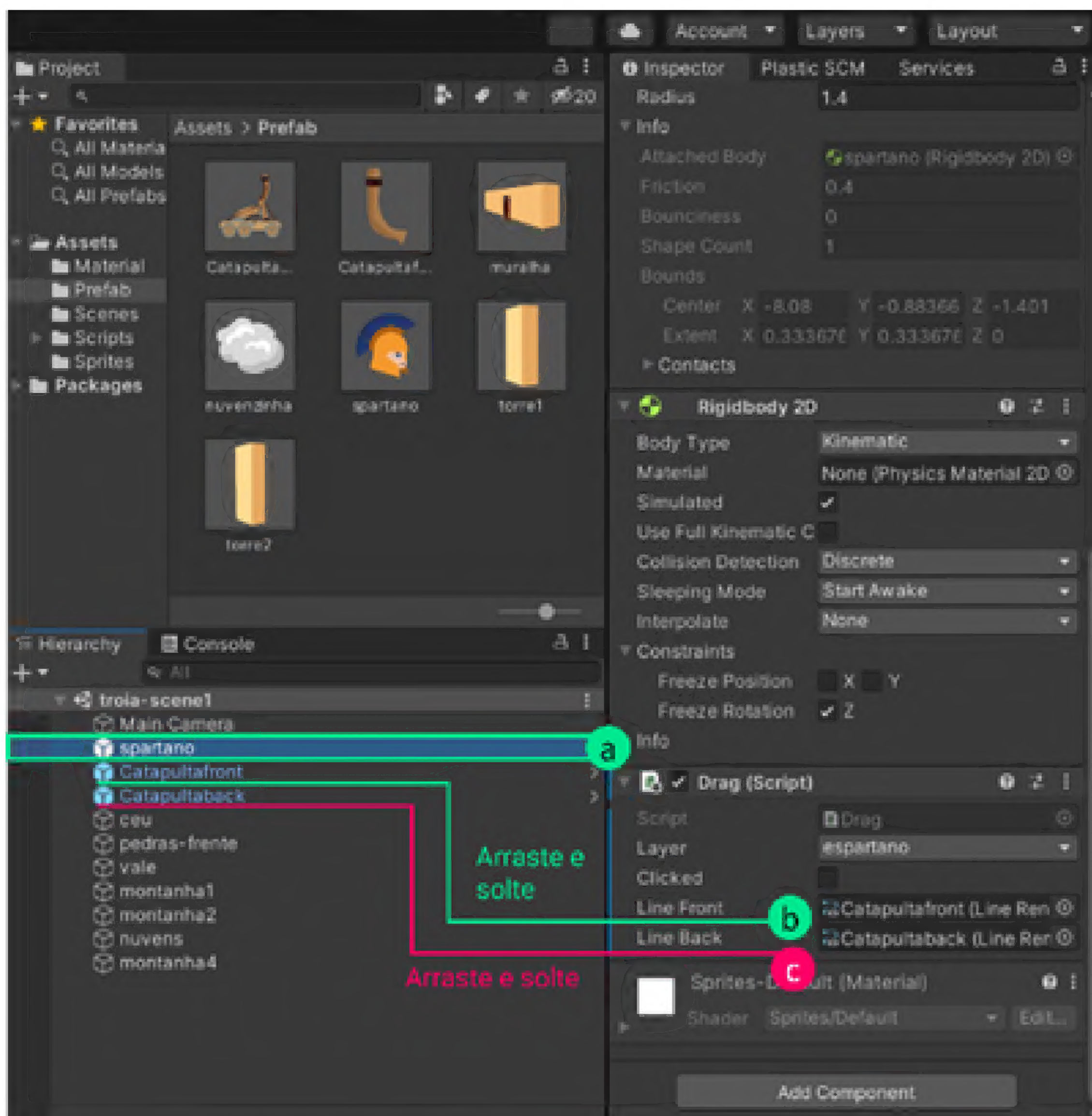


Figura 6 – Vinculação de Line Renderer | Autor, 2022

Com esse código já será possível você ver o efeito elástico (Figura 7) acontecendo, porém, será necessário trabalhar a estética e o acabamento do elástico da catapulta. Para esse fim criei outras variantes para que a linha se estendesse até a extremidade do CircleCollider2D vinculado com o Prefab do

Personagem. No final desse material deixarei o link para vocês baixarem o código.



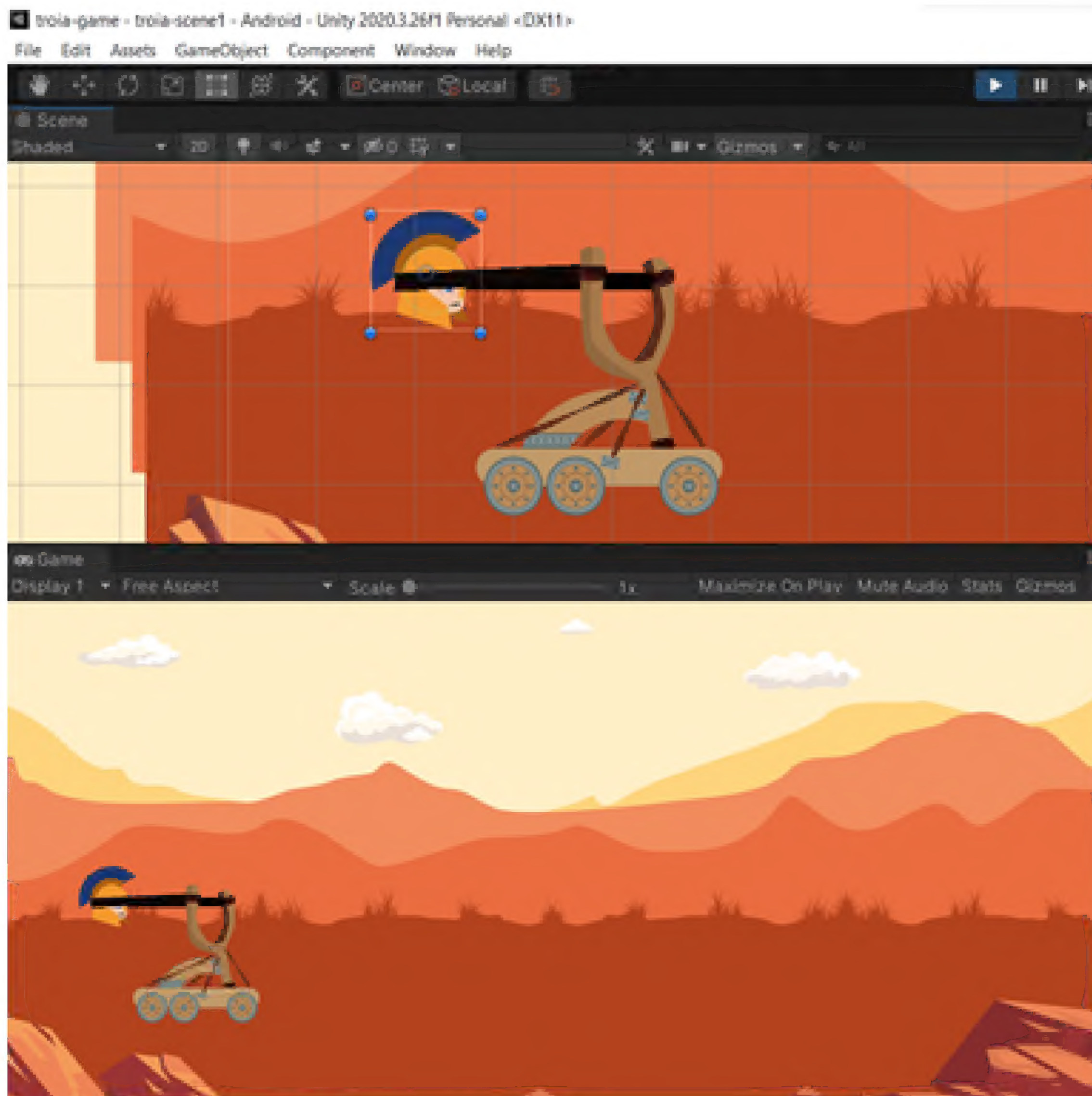


Figura7 – Efeito Elástico | Autor, 2022

2. Efeito Catapulta

Com o objetivo de criar o efeito físico para que o personagem seja lançado, precisaremos adicionar na catapulta mestra um componente “Physics 2D > Rigidbody 2D” sendo que o seu *Bory Tipe* deverá ser *Kinematic*. Depois, precisaremos adicionar o componente “Physics 2D > Spring Joint 2D” no personagem para que possamos atirá-lo. No Inspector você precisará (Figura 8):

a. Selecionar o Personagem em Hierarchy;

b. Arrastar a Catapulta Mestra de Hierarchy para o "Spring Joint 2D no elemento Connected Ring Body, do personagem;

c. Desconectar o item "Auto Configure Distance";

d. Delimitar a distância para "1" e;

e. Frequência para "3".

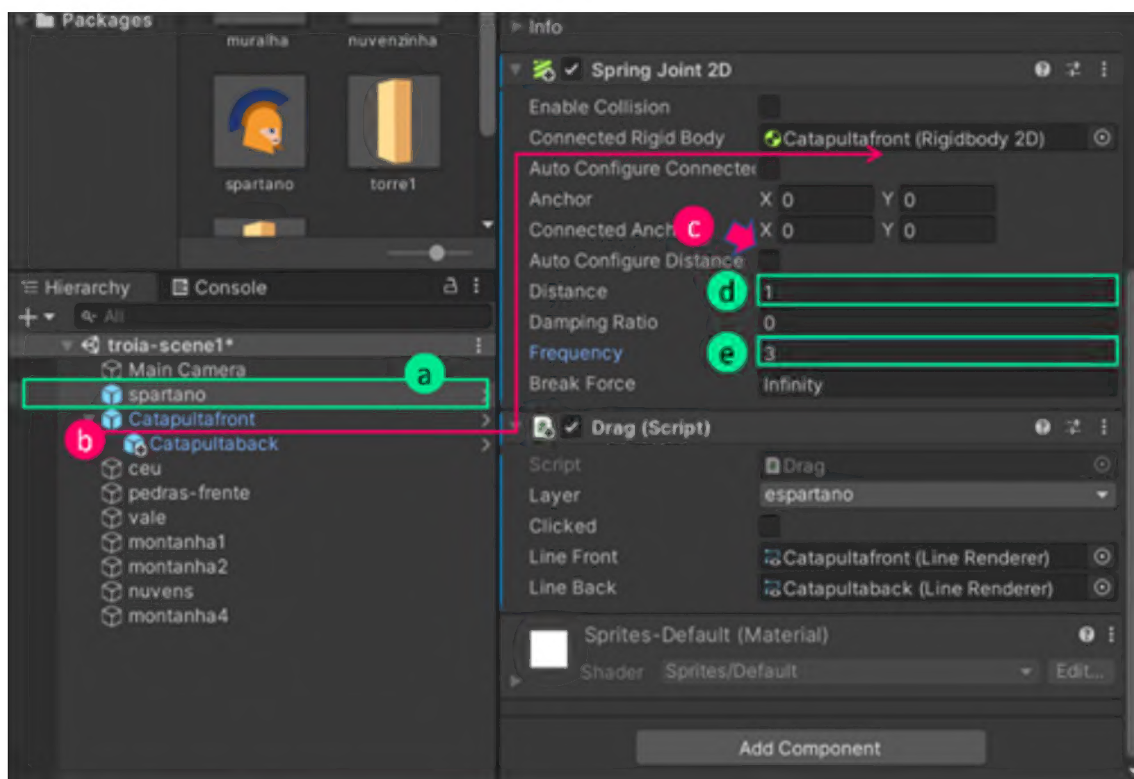


Figura 8 – Spring Joint 2D | Autor, 2022

Vamos então adicionar alguns elementos em nosso código, abra o código C# para fazer a seguinte implementação de variáveis (Figura 9):

- **private SpringJoint2D spring;** declarando o nome do efeito dentro do código.
- **private Vector2 preVelox;** declarando um vetor de velocidade.

- **private Rigidbody2D espartanoRB;** declarando o componente RigidBody2D lincado no personagem.



```
//variáveis para o personagem ser movido pelo touch
private Collider2D drag;
public LayerMask layer;
[SerializeField]
private bool clicked;
private Touch touch;

//variáveis para o comportamento da linha (elástico) da catapulta

public LineRenderer lineFront;
public LineRenderer lineBack;

private Ray leftCatapultRay;
private CircleCollider2D espartanoCol;
private Vector2 catapultToSparta;
private Vector3 pointL;

//variáveis para atirar o personagem
private SpringJoint2D spring;
private Vector2 preVelox;
private Rigidbody2D espartanoRB;
```

Figura 9 – Variáveis para o lançamento de personagem pela catapulta | Autor, 2022

Agora em "Start" criar os objetos que serão responsáveis pelo efeito atirar o personagem, são eles "spring = GetComponent<SpringJoint2D>();" e "espartanoRB = GetComponent<Rigidbody2D>();".

Em seguida você deverá declarar um falso para o efeito Kinemastic para que o personagem seja solto após a declaração de "Cliked = false", ou seja, precisará colocar o seguinte código: "espartanoRB.isKinematic = false;".

Mas para que o código de arremesso do personagem esteja completo você precisará acrescentar o seguinte método:

```
// médoto para catapultar personagem
1 referência
void SpringEffect()
{
    if(spring != null)
    {
        if (espartanoRB.isKinematic == false)
            if (preVelox.sqrMagnitude > espartanoRB.velocity.sqrMagnitude)
            {
                lineFront.enabled = false;
                lineBack.enabled = false;
                Destroy(spring);
                espartanoRB.velocity = preVelox;
            }
    }
}
```

Figura 10 – Método para o arremesso do personagem | Autor, 2022

3. Efeito Câmera Segue

Uma vez que a mecânica e ações do personagem foram desenvolvidas para dar a impressão de estilingue, precisaremos configurar o chão para que o personagem quique ao bater em uma superfície, além de que precisaremos fazer que a câmera siga a ação.

Para resolver a questão do chão não é complicado, precisaremos criar um GameObject vazio e adicionar o componente EdgeCollider2D e ativando os pontos vetoriais redimensioná-lo para o tamanho do chão.

Para continuar o efeito da câmera seguindo o percurso do personagem você precisará criar dois GameObject no Unity um como nomeado como lado esquerdo, outro como lado direito. Depois um código C# que será aplicado à câmera (Figura 11). Em seguida em câmera vincular os GameObjects Lado Direito, Lado Esquerdo e Spartano.

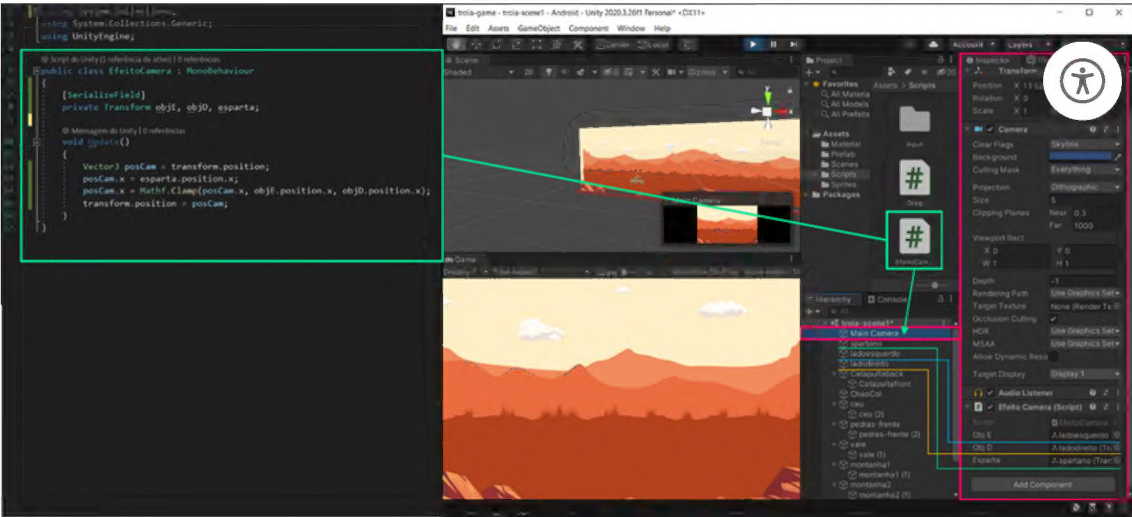


Figura 11 – Efeito de Câmera Acompanhando a Ação

4. Morte do Personagem

Para desenvolver um meio para categorizar a morte do personagem, precisaremos trabalhar no código um método que seja acionado a partir da desaceleração do personagem. Portanto, o código precisa estar no Spartano por meio do C# Drag.

Para tanto, você pode fazer uso de um código simples ou de um com a utilização de efeito com partículas deixando a morte do personagem mais interessante visualmente (Tabela 2).

Simples	Com Efeito Partícula
void MataPersonagem () { if(espartanoRB.velocity.magnitude < 0.5f) { StartCoroutine(TempoMorte()); } IEnumerator TempoMorte() { yield return new WaitForSeconds(4); Destroy(gameObject); } }	void MataPersonagem () { if(espartanoRB.velocity.magnitude == 0 || espartanoRB.IsSleeping()) { StartCoroutine(TempoMorte()); } IEnumerator TempoMorte() { yield return new WaitForSeconds(4); Instantiate (bomb, new Vector2(transform.position.x, transform.position.y), Quaternion.identity); Destroy(gameObject); } }

Tabela 2 – Códigos de morte do personagem

No caso do código mais simples, o personagem simplesmente sumirá da tela. Vale ressaltar que o nome do método, em ambos os casos, precisará ser

declarado depois do clicked (Figura 12).



```
}
if (touch.phase == TouchPhase.Ended || touch.phase == TouchPhase.Canceled)
{
    //ao soltar o personagem ele deixará de ser kinematic
    espartanoRB.isKinematic = false;
    clicked = false;

    MataPersonagem();
}
```

Figura 12 – Declaração do Void MataPersonagem | Autor, 2022

No caso de se optar pelo efeito partícula, você deverá fazer o efeito, torná-lo parte filho de um GameObject vazio e salvá-los como Prefab. Depois é só vinculá-lo ao personagem por meio do Unity (Figura 13).

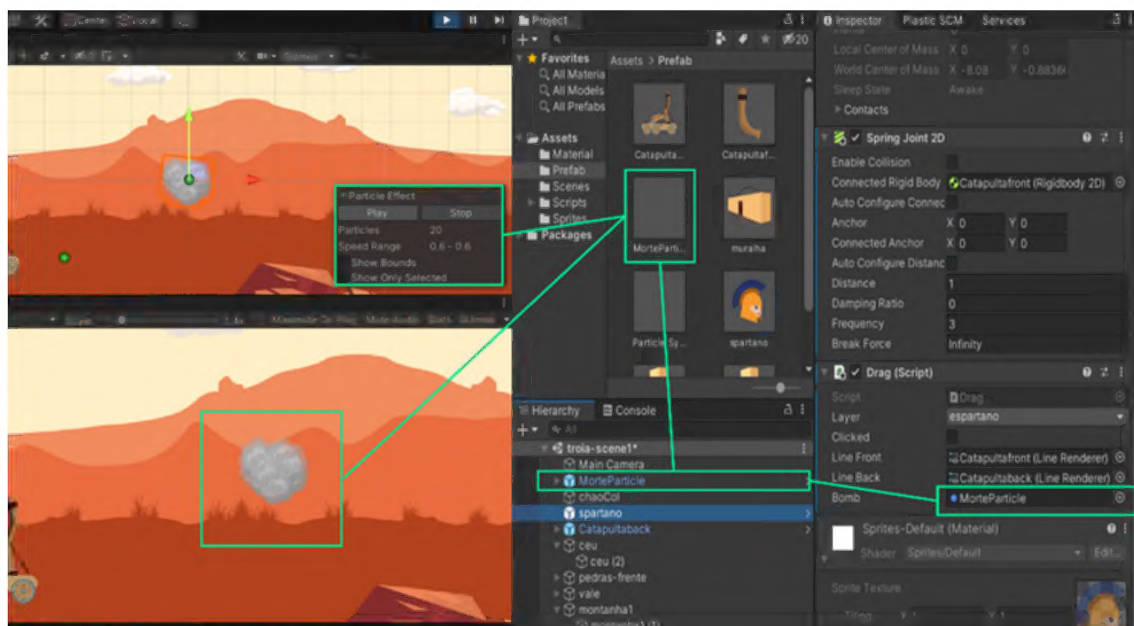


Figura 13 – Vinculando o efeito de partículas

Espero que tenha gostado até aqui, na próxima veremos como limitar o elástico e criar os obstáculos para serem destruídos. Até lá… Vida longa e próspera.

Atividade extra



Baixe o jogo Angry Bird em seu celular e jogue. Angry Birds é um jogo eletrônico do estilo puzzle desenvolvido por Rovio Entertainment. O jogo foi lançado em 2009 e foi muito utilizado, além de diversão, para o ensino de física.

Disponível em: <https://play.google.com/store/search?q=angry%20birds>. Acesso em 07 fev. 2022.

Leia o artigo:

PINTO, A. F.; et al. O potencial do jogo angry birds para o ensino de física. Caderno de Física N. 18. Feira de Santana: UEFS, 2020. p. 1201.1-1201.11. Disponível em: <http://dfisweb.uefs.br/caderno/vol18n1/S2_Artigo01_Arturlara_%20Jogo_Angry_Birds.pc

Se possível tente, ao jogar, fazer a relação da sua experiência com o relato do artigo.

Referência Bibliográfica

NASCIMENTO, W. **Jogos 2D com Unity e C#**. São Paulo: Udemy, 2022.

Ir para exercício